UTC #168 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2021-jul-20

Properties & algorithms

We are a group of Unicode contributors who take an interest in properties and algorithms.
We look at relevant feedback reports and documents that Unicode receives, do some research, and submit UTC documents with recommendations as input to UTC meetings.

This group started with the UCD file and production tool maintainers, and with Markus Scherer as the chair. Several UTC participants have requested and received invitations to join. We discuss via email, shared documents, and sometimes video meetings.

Participants

The following people have contributed to this document:

Markus Scherer (chair), Asmus Freytag, Ken Whistler, Peter Constable, Ned Holbrook, Mark Davis, Roozbeh Pournader

Unicode 14 beta

https://www.unicode.org/review/pri433/
(Ignoring here feedback about Unihan, emoji, and annotations.)

β1 provisional named sequences

Recommended UTC actions

1. Approve the 19 named sequences in Unicode 13.0 NamedSequencesProv.txt, for Unicode 14.
2. Action item for Ken Whistler: Move the 19 named sequences from NamedSequencesProv.txt to NamedSequences.txt, for Unicode 14.

Feedback (verbatim)

From Ken:

In prepping alpha UCD data files, I came up against the fact that we still have 19 provisional named sequences sitting around for various Indic script precomposed characters with nuktas. These were accepted provisionally back in January 2020:

https://www.unicode.org/L2/L2020/20015.htm#162-C22

but we don't have any follow up yet for 14.0 to make them approved final named sequences. It would be a good idea not to leave these in limbo indefinitely, so I suggest you add an item to the agenda for the next properties & algorithms group meeting, to make sure we get a recommendation to approve these into the report.

Background information / discussion

https://www.unicode.org/Public/UCD/latest/ucd/NamedSequencesProv.txt

```
# Entries that correspond to Indic characters with nuktas
# that are also listed in CompositionExclusions.txt.
# These characters decompose for normalized text, even
# in NFC. Having named sequences for these helps in
# certain specifications, including Label Generation Rules (LGR)
# for Internationalized Domain Names (IDN).
#
# Provisional 2020-01-16

DEVANAGARI SEQUENCE FOR LETTER QA; 0915 093C
DEVANAGARI SEQUENCE FOR LETTER KHHA; 0916 093C
DEVANAGARI SEQUENCE FOR LETTER GHHA; 0917 093C
DEVANAGARI SEQUENCE FOR LETTER ZA; 091C 093C
DEVANAGARI SEQUENCE FOR LETTER DDDHA; 0921 093C
DEVANAGARI SEQUENCE FOR LETTER RHA; 0922 093C
DEVANAGARI SEQUENCE FOR LETTER FA; 092B 093C
DEVANAGARI SEQUENCE FOR LETTER YYA; 092F 093C
BENGALI SEQUENCE FOR LETTER RRA; 09A1 09BC
BENGALI SEQUENCE FOR LETTER RHA; 09A2 09BC
BENGALI SEQUENCE FOR LETTER YYA; 09AF 09BC
GURMUKHI SEQUENCE FOR LETTER LLA; 0A32 0A3C
GURMUKHI SEQUENCE FOR LETTER SHA; 0A38 0A3C
GURMUKHI SEQUENCE FOR LETTER KHHA; 0A16 0A3C
GURMUKHI SEQUENCE FOR LETTER GHHA; 0A17 0A3C
GURMUKHI SEQUENCE FOR LETTER ZA; 0A1C 0A3C
GURMUKHI SEQUENCE FOR LETTER FA; 0A2B 0A3C
ORIYA SEQUENCE FOR LETTER RRA; 0B21 0B3C
ORIYA SEQUENCE FOR LETTER RHA; 0B22 0B3C
```

β2 Other_Lowercase Property for New Modifier Letters

Recommended UTC actions

1. Assign Other_Lowercase to the modifier letters in Latin Extended-F recommended by Charlotte: 10780, 10783..10785, 10787..107B0, 107B2..107B5, 107BA; and also to 107B6..107B9; for Unicode 14.
2. Action item for Ken Whistler: Assign Other_Lowercase to the modifier letters in Latin Extended-F listed in L2/21-126 item β2, for Unicode 14.
3. Action item for Ken Whistler: Review existing modifier letters encoded for phonetic transcriptions and recommend which ones should be assigned Other_Lowercase, such as possibly U+AB69 MODIFIER LETTER SMALL TURNED W, for Unicode 14.

Feedback (verbatim)

Date/Time: Fri Jun 11 12:50:02 CDT 2021
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: PRI #433: Other_Lowercase Property for New Modifier Letters

The following characters in the new Latin Extended-F block should be given the property Other_Lowercase=True for consistency with other similar modifier letters already encoded:

U+10780 (MODIFIER LETTER SMALL CAPITAL AA)
U+10783..U+10785 (MODIFIER LETTER SMALL AE..MODIFIER LETTER SMALL B WITH HOOK)
U+10787..U+107B0 (MODIFIER LETTER SMALL DZ DIGRAPH..MODIFIER LETTER SMALL V WITH RIGHT HOOK)
U+107B2..U+107B5 (MODIFIER LETTER SMALL CAPITAL Y..MODIFIER LETTER BILABIAL CLICK)
U+107BA (MODIFIER LETTER SMALL S WITH CURL)

It is unclear whether the following characters should be classified as lowercase as well since their base forms are Other_Letter rather than Lowercase_Letter:

U+10781..U+10782 (MODIFIER LETTER SUPERSCRIPT TRIANGULAR COLON..MODIFIER LETTER SUPERSCRIPT HALF TRIANGULAR COLON)
U+107B6..U+107B9 (MODIFIER LETTER DENTAL CLICK..MODIFIER LETTER RETROFLEX CLICK WITH RETROFLEX HOOK)

Background information / discussion

https://raw.githubusercontent.com/unicode-org/icu/main/icu4c/source/data/unidata/ppucd.txt

```
block;10780..107BF;age=14.0;Alpha;blk=Latin_Ext_F;CI;CWKCF;Dia;dt=Sup;gc=Lm;Gr_Base;IDC;ID
S;lb=AL;NFKC_QC=N;NFKD_QC=N;SB=LE;sc=Latn;WB=LE;XIDC;XIDS
# 10780..107BF Latin Extended-F
# Modifier letter for VoQS
cp;10780;-CWKCF;dt=None;na=MODIFIER LETTER SMALL CAPITAL AA;NFKC_QC=Y;NFKD_QC=Y
# Modifier letters for IPA
cp;10781;dm=02D0;Ext;na=MODIFIER LETTER SUPERSCRIPT TRIANGULAR COLON;NFKC_CF=02D0
cp;10782;dm=02D1;Ext;na=MODIFIER LETTER SUPERSCRIPT HALF TRIANGULAR COLON;NFKC_CF=02D1
cp;10783;dm=00E6;na=MODIFIER LETTER SMALL AE;NFKC_CF=00E6
cp;10784;dm=0299;na=MODIFIER LETTER SMALL CAPITAL B;NFKC_CF=0299
cp;10785;dm=0253;na=MODIFIER LETTER SMALL B WITH HOOK;NFKC_CF=0253
unassigned;10786
cp;10787;dm=02A3;na=MODIFIER LETTER SMALL DZ DIGRAPH;NFKC_CF=02A3
cp;10788;dm=AB66;na=MODIFIER LETTER SMALL DZ DIGRAPH WITH RETROFLEX HOOK;NFKC_CF=AB66
…
```

Compared with

```
block;A720..A7FF;age=5.1;Alpha;blk=Latin_Ext_D;Cased;CWCM;gc=Ll;Gr_Base;IDC;IDS;lb=AL;SB=L
O;sc=Latn;WB=LE;XIDC;XIDS
```

```
# A720..A7FF Latin Extended-D
…
# Additions for Extended IPA
cp;A7F8;age=6.1;CI;-CWCM;CWKCF;Dia;dm=0126;dt=Sup;FC_NFKC=0127;gc=Lm;Lower;na=MODIFIER
LETTER CAPITAL H WITH STROKE;NFKC_CF=0127;NFKC_QC=N;NFKD_QC=N
cp;A7F9;age=6.1;CI;-CWCM;CWKCF;Dia;dm=0153;dt=Sup;gc=Lm;Lower;na=MODIFIER LETTER SMALL
LIGATURE OE;NFKC_CF=0153;NFKC_QC=N;NFKD_QC=N

block;AB30..AB6F;age=7.0;Alpha;blk=Latin_Ext_E;Cased;gc=Ll;Gr_Base;IDC;IDS;lb=AL;Lower;SB=
LO;sc=Latn;WB=LE;XIDC;XIDS
# AB30..AB6F Latin Extended-E
…
# Modifier letters for German dialectology
cp;AB5B;-Alpha;-Cased;CI;Dia;gc=Sk;-IDC;-IDS;-Lower;na=MODIFIER BREVE WITH INVERTED
BREVE;SB=XX;sc=Zyyy;-XIDC;-XIDS
cp;AB5C;CI;CWKCF;Dia;dm=A727;dt=Sup;gc=Lm;na=MODIFIER LETTER SMALL
HENG;NFKC_CF=A727;NFKC_QC=N;NFKD_QC=N
cp;AB5D;CI;CWKCF;Dia;dm=AB37;dt=Sup;gc=Lm;na=MODIFIER LETTER SMALL L WITH INVERTED LAZY
S;NFKC_CF=AB37;NFKC_QC=N;NFKD_QC=N
cp;AB5E;CI;CWKCF;Dia;dm=026B;dt=Sup;gc=Lm;na=MODIFIER LETTER SMALL L WITH MIDDLE
TILDE;NFKC_CF=026B;NFKC_QC=N;NFKD_QC=N
cp;AB5F;CI;CWKCF;Dia;dm=AB52;dt=Sup;gc=Lm;na=MODIFIER LETTER SMALL U WITH LEFT
HOOK;NFKC_CF=AB52;NFKC_QC=N;NFKD_QC=N
…
# Letters for Scots dialectology
cp;AB68;age=13.0;na=LATIN SMALL LETTER TURNED R WITH MIDDLE TILDE
cp;AB69;age=13.0;-Cased;CI;CWKCF;Dia;dm=028D;dt=Sup;gc=Lm;-Lower;na=MODIFIER LETTER SMALL
TURNED W;NFKC_CF=028D;NFKC_QC=N;NFKD_QC=N;SB=LE
```

Ken:

- For precedents, look at Other_Lowercase in PropList.txt.
- Modifier letters generally get Lowercase because they are used in IPA and other phonetic transcriptions in a conceptually-lowercase context.
- Important: This is determined by usage, not by shape.
- Some characters are modifiers of modifiers. For example, U+10781 MODIFIER LETTER SUPERSCRIPT TRIANGULAR COLON which is a superscripted length mark (as is U+10782). As such, neither it nor the character it decomposes to (U+02D0) are Lowercase. Contrast with other modifiers that behave more like letters, for example U+107B6 MODIFIER LETTER DENTAL CLICK.

β3 3 Name Defects in Ethiopic Extended-B Tables

Recommended UTC actions

1. FYI: The names of these three characters have already been changed in the data files. The corrected names are those that the UTC has approved before. The derived files have been regenerated for the names in comments to match.

Feedback (verbatim)

Date: Mon, 5 Jul 2021 12:19:48 -0400
Name: Daniel Yacob
Subject: 3 Name Defects in Ethiopic Extended-B Tables

I was just working with the table for the Ethiopic Extended-B range, published under the U14 Beta delta listing here: https://www.unicode.org/charts/PDF/Unicode-14.0/

I found that a few names were off, I believe the error originates from the UniBook output that I submitted earlier this year. The defects are:

1E7E9 ETHIOPIC SYLLABLE HWI
1E7EA ETHIOPIC SYLLABLE HWEE
1E7EB ETHIOPIC SYLLABLE HWE

In each case the name base "H" should have been "HH", the corrected names:

1E7E9 ETHIOPIC SYLLABLE HHWI
1E7EA ETHIOPIC SYLLABLE HHWEE
1E7EB ETHIOPIC SYLLABLE HHWE

I apologize for this. The names are correct in our proposal L2/21-037 (https://www.unicode.org/L2/L2021/21037-gurage-adds.pdf) and I think the difference simply stems from a typographical error that I made while working with UniBook.

thank you,

-Daniel

Background information / discussion

Confirmed clerical error.

β4 ORNATE LEFT PARENTHESIS should be Ps ?

Recommended UTC actions

1. No change

Feedback (verbatim)

Date/Time: Thu Jul 8 20:01:46 CDT 2021
Name: philip r brenan
Report Type: Error Report

Opt Subject: ORNATE LEFT PARENTHESIS should be Ps ?

FD3E;ORNATE LEFT PARENTHESIS;Pe;0;ON;;;;;N;;;;;
FD3F;ORNATE RIGHT PARENTHESIS;Ps;0;ON;;;;;N;;;;;

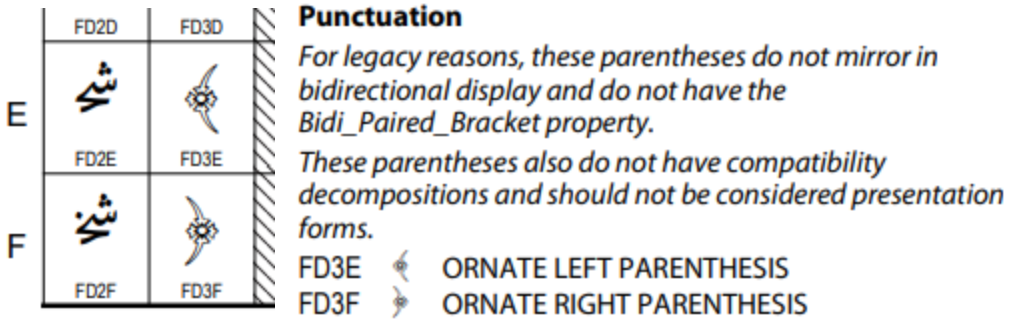
Possibly the Pe and Ps are the wrong way around?

Background information / discussion

https://www.unicode.org/reports/tr44/#General_Category_Values

Ps	Open_Punctuation	an opening punctuation mark (of a pair)
Pe	Close_Punctuation	a closing punctuation mark (of a pair)

https://www.unicode.org/charts/PDF/UFB50.pdf



Roozbeh Pournader: The present UCD values appear correct, esp. since these don't mirror and are most commonly used in RTL text.

Peter Constable: A quirk with FD3E is that it can immediately *precede* in logical order the character that is visually at the end of the parenthetic block. And a quirk with FD3F is that it can logically follows the character that starts the block.

More generally, the logical placement within a string and the visual appearance can be quite different. E.g., logically they might be just a few characters apart while visually they can be many more characters apart. (Which is why they can't be Bidi_Paired_Bracket.)

There's no question that, in reading, FD3E ends a parenthetic block, and FD3F begins a block. But you can't use the Ps/Pe properties to predict where they would occur in a string or to identify semantically what is a parenthetic sub-string within the text without running BiDi Alg.

Swapping the Ps/Pe values for these characters would not make sense. Changing both to Po, though, perhaps might make sense?

β5 BidiMirroring.txt “BEST FIT” for ∉ ∌

Recommended UTC actions

1. Action item for Ken Whistler: In BidiMirroring.txt, add the “best fit” comment to U+2209 NOT AN ELEMENT OF and U+220C DOES NOT CONTAIN AS MEMBER, for Unicode 14.

Feedback (verbatim)

Date/Time: Tue Jul 13 16:02:19 CDT 2021
Name: Kent Karlsson
Report Type: Error Report
Opt Subject: BidiMirroring.txt

∉ ∌ # NOT AN ELEMENT OF
∌ ∉ # DOES NOT CONTAIN AS MEMBER

These should get the annotation [BEST FIT].

Background information / discussion

https://www.unicode.org/Public/UCD/latest/ucd/BidiMirroring.txt
A comment indicates where the characters are "BEST FIT" mirroring.

2208; 220B # ELEMENT OF
2209; 220C # NOT AN ELEMENT OF
220A; 220D # SMALL ELEMENT OF
220B; 2208 # CONTAINS AS MEMBER
220C; 2209 # DOES NOT CONTAIN AS MEMBER

2268; 2269 # [BEST FIT] LESS-THAN BUT NOT EQUAL TO
2269; 2268 # [BEST FIT] GREATER-THAN BUT NOT EQUAL TO

Markus: If one of the glyphs commonly has a slanted strike-through, then graphic mirroring will slant it the wrong way. We write “best fit” if graphic mirroring yields readable but sub-optimal results.

Documents

D1: Requirements and Process for Changing Script Status for Identifier Use

L2/21-030r2 from Asmus Freytag

Proposal-Questionnaire-Identifier-Type feedback doc from Mark Davis

Recommended UTC actions

1. Brief discussion of document L2/21-030.
2. Action item for Asmus Freytag, Mark Davis, and the Properties & Algorithms group to take the guidelines in document L2/21-030 to develop guidelines for proposals to change the status of scripts in UAX #31.

Summary

Doc intro:

> UTS#39 defines Identifier type Recommended as characters in “widespread common everyday use”. Formally, the definition is based on membership of the character in a Recommended script in UAX#31 (with some exceptions). Recommended scripts are therefore in “widespread common everyday use”, while other scripts with less active modern use might be classed as Limited_Use.
>
> These characterizations are not permanent; they are intended to track actual use of a given script, including any significant changes in usage. The definition of Recommended script is used as input to other specifications outside of the Unicode Standard, such as the Label Generation Rules for the DNS Root Zone (see “Root Zone LGR” under https://icann.org/idn for details).
>
> Because of such dependencies, it is advisable to use a very deliberate process when adjusting the status of a script in UAX#31 (and therefore the Identifier_Type of its member characters). Such a process must first and foremost establish whether the usage for a Limited_Use script has changed sufficiently so that it fits the requirements of being in “widespread common everyday use”.

Followed by a discussion of suggested criteria.

Background information / discussion

We discussed this topic, and iterated with Asmus on his document 21-030.

D2: Better default values in the UCD

L2/21-100 from Markus Scherer & Mark Davis

Recommended UTC actions

1. Explicitly allow multiple @missing lines in UCD files, with last-line-wins semantics, as proposed in L2/21-100 part 1.
2. Action item for Markus Scherer, Mark Davis, and the editorial committee: Document in UAX #44 that data files may contain multiple @missing lines, with last-line-wins semantics; for Unicode 15; see L2/21-126 item D2.
3. Action item for Markus Scherer and Mark Davis: Change one or more UCD files to use multiple @missing lines; for Unicode 15; see L2/21-126 item D2.

Summary

Doc intro:
The Unicode Character Database files specify both the default values of properties, and the specific values for specific code points and characters. Some properties have different default values for different blocks, or for certain ranges set aside for various types of future allocations. Default values are specified in special comments at the head of the file containing an @missing directive (see the “Status Quo” below for details).

Summary of proposal:
1. Explicitly specify behavior with multiple @missing lines
    # @missing: 0600..07BF; Arabic_Letter
2. Allow block values in the place of code point ranges of @missing lines
    # @missing: Block=Arabic; Arabic_Letter

Background information / discussion

Favorable discussion about part 1 (for consideration for 15.0). Suggestion to add a comment-in-comment with the Block name or what script the range is reserved for (not machine-readable). For example:

```
# @missing: 0600..07BF; Arabic_Letter # Block=Arabic
# @missing: 0870..089F; Right_To_Left # tentatively reserved for Quranic additions
```

Unfavorable discussion about part 2.

D3: Pamudpod properties (Tagalog and Hanunoo)

L2/21-117 from Roozbeh Pournader

Recommended UTC actions

1. We agree with the SAH recommendation.
2. Action for Mark Davis and the Properties & Algorithms group to create a checklist to verify that proposed changes to properties do not cause problems for identifiers (UAX #31) or for IDNA (UTS #46). This should be covered by an “invariant test” if possible.

Summary

Align General Category, Bidi Class, and Indic Positional Category of old U+ 1734 HANUNOO SIGN PAMUDPOD and new U+1715 TAGALOG SIGN PAMUDPOD, changing properties of both.

Background information / discussion

Debbie sent this to Markus:

We reviewed this short document about the character properties for the Tagalog and Hanunoo *pamudpod* characters. Review of the properties by Roozbeh Pournader uncovered discrepancies between the two.

After discussion, the group agreed that the general category for both characters should be Mc, the bidi property L, and the IndicPositionalCategory should be changed to Right. (Note: The document recommends

HANUNOO SIGN PAMUDPOD retain its current bidi class, NSM, but the group agreed that L was appropriate.)

We recommend the UTC approve the following adjustments:

U+1734 HANUNOO SIGN PAMUDPOD:

Change the gc from Mn to Mc

Change the bidi class from NSM to L

Change the IndicPositionalCategory from Bottom to Right.

U+1715 TAGALOG SIGN PAMUDPOD:

Change the IndicPositionalCategory Bottom_and_Right to Right.

The following summarizes the corrected UnicodeData and IndicPositionalCategory entries:

UnicodeData.txt:

1715;TAGALOG SIGN PAMUDPOD;Mc;9;L;;;;;N;;;;;

1734;HANUNOO SIGN PAMUDPOD;Mc;9;L;;;;;N;;;;;

IndicPositionalCategory.txt:

1715 ; Right

1734 ; Right

(Document has been posted as L2/21-117 with no changes to the version seen by SAH.)

Public Review Issues

https://www.unicode.org/review/

PRI #417: Proposed Update UAX #29, Unicode Text Segmentation

https://www.unicode.org/review/pri417/

PRI417a: Unicode Standard Annex #29 - 3 Grapheme Cluster Boundaries - SpacingMark

Same feedback as L2/21-069R item PRI417b. New information for UTC #168.

Recommended UTC actions

1. Remove the assignment of GCB=SpacingMark for U+11720..11721 AHOM VOWEL SIGN A & AA, letting them default to GCB=Other.
2. Action item for Mark Davis and Markus Scherer: Remove the assignment of GCB=SpacingMark for U+11720..11721 AHOM VOWEL SIGN A & AA, letting them default to GCB=Other. For Unicode 14. Reference: L2/21-xxx item PRI417a.
3. Action item for Mark Davis and the editorial committee: Add a note in UTS #39 like the one in UAX #14 about data files being normative.

Feedback (verbatim)

Date/Time: Fri Jan 29 18:14:29 CST 2021
Contact: johnsoneal@gmail.com
Name: Neal Johnson
Report Type: Error Report
Opt Subject: Unicode Standard Annex #29 - 3 Grapheme Cluster Boundaries - SpacingMark

Unicode Standard Annex #29 - 3 Grapheme Cluster Boundaries - SpacingMark (https://www.unicode.org/reports/tr29/#SpacingMark) states that U+11720 and U+11721 should be specifically excluded. However "GraphemeBreakProperty.txt" list both as included and as such {{UCharacter.getIntPropertyValue(0x11721, UProperty.GRAPHEME_CLUSTER_BREAK) }} return 10 "SPACING_MARK".

I am not sure if this an issue in the "GraphemeBreakProperty.txt" data file or an issue in Annex #29.

(submitted by Markus on behalf of Neal who mis-reported this as https://unicode-org.atlassian.net/browse/ICU-21438)

Background information / discussion

https://www.unicode.org/reports/tr29/#SpacingMark

SpacingMark	Grapheme_Cluster_Break ≠ Extend, *and* General_Category = Spacing_Mark, *or* *any of the following (which have* General_Category = Other_Letter*):* U+0E33 (◌ำ) THAI CHARACTER SARA AM U+0EB3 (◌ຳ) LAO VOWEL SIGN AM *Exceptions: The following (which have* General_Category = Spacing_Mark *and would otherwise be included) are specifically excluded:* U+102B (◌ါ) MYANMAR VOWEL SIGN TALL AA ... U+AA7B (◌ꩻ) MYANMAR SIGN PAO KAREN TONE U+AA7D (◌ꩽ) MYANMAR SIGN TAI LAING TONE-5 U+11720 (◌𑜠) AHOM VOWEL SIGN A U+11721 (◌𑜡) AHOM VOWEL SIGN AA

https://www.unicode.org/Public/UCD/latest/ucd/auxiliary/GraphemeBreakProperty.txt

```
116AE..116AF  ; SpacingMark # Mc   [2] TAKRI VOWEL SIGN I..TAKRI VOWEL SIGN II
116B6         ; SpacingMark # Mc       TAKRI SIGN VIRAMA
11720..11721  ; SpacingMark # Mc   [2] AHOM VOWEL SIGN A..AHOM VOWEL SIGN AA
11726         ; SpacingMark # Mc       AHOM VOWEL SIGN E
1182C..1182E  ; SpacingMark # Mc   [3] DOGRA VOWEL SIGN AA..DOGRA VOWEL SIGN II
11838         ; SpacingMark # Mc       DOGRA SIGN VISARGA
```

The Ahom vowel signs A & AA were added to the UAX #29 exceptions list for SpacingMark (Table 2) in draft tr29-26.html with this review note:

Review Note:
The additional exceptions proposed for Ahom [L2/12-309R] were inferred by extrapolating the former list of exclusions to the new repertoire in Unicode 8.0, following the rationale given in [L2/11-114] which was applied in [tr29-18.html] for Unicode 6.1. Careful review is advised. If the proposed exceptions are accepted, then U+11720..U+11721 will also be excluded from the list of SpacingMark characters in GraphemeBreakProperty.txt for Unicode 8.0.

References:
[L2/12-309R] Martin Hosken, Stephen Morey, *Revised Proposal to add the Ahom Script in the SMP of the UCS*
[L2/11-114] Martin Hosken, *Proposal to change grapheme extending properties of various characters*
[tr29-18.html] Mark Davis, *Proposed Update UAX #29 for Unicode 6.1*

Excluding these characters from SpacingMark means that their WB values change to Other.

Recommended UTC action: AI: Remove the assignment of GCB=SpacingMark for U+11720..11721 AHOM VOWEL SIGN A & AA, letting them default to GCB=Other. For Unicode 14. Reference: L2/21-069 item P417b.

Peter: On investigation I believe it is correct to change the data to exclude 11720..11721 from SPACING_MARK.

Peter Constable 2021-may-24

Debbie and I followed up with Stephen Morey regarding the issue of whether Ahom vowels A and AA should have the SpacingMark property. The conclusion is that they should not. [In other words, the proposed exception should be kept. Excerpts from the email thread of that discussion follow.]

From: Peter Constable
Sent: May 24, 2021 9:47 AM
To: Stephen Morey; Martin Hosken
Cc: Deborah Anderson
Subject: RE: Ahom vowels A/AA and SpacingMark property

Thanks, Stephen and Martin.

The feedback I've heard from both of you is that the Ahom vowels A/AA should not have the SpacingMark property so that they have comparable cluster behaviour to the Thai example. Note that this isn't permanent: if it's later determined that a different cluster behaviour is preferable, this property value can be changed.

Peter

From: Stephen Morey
Sent: May 24, 2021 2:19 AM
To: Martin Hosken
Cc: Peter Constable; Deborah Anderson

Dear Everyone

Thanks very much, Martin, for your clear email, and discussion of so many of the issues. I'm talking to community members in the Tai communities on an increasingly regular basis about all these issues and it's taking a lot of time – such as the continuing desire to have 'disunification' of the 'dotted forms' of Tai Phake and Khamti &c because the variant selector forms are so difficult to access. Martin's point about the continuing need for research is important and I wish I could spend more of my time researching these Tai scripts but …

Meanwhile I am glad that Martin agrees with my rather impressionistic feeling that Ahom should be treated like Thai! I could raise this as an issue on a Facebook post, but I suspect won't fully understand the implications of it, any more than I fully do!

All the best

Stephen

From: Martin Hosken
Sent: Monday, 24 May 2021 12:44 PM
To: Stephen Morey
Cc: Peter Constable; Deborah Anderson
Subject: Re: Ahom vowels A/AA and SpacingMark property

Dear Peter, et al.

The SpacingMark property is designed to reflect desired usage and not the other way around. Hence your asking the question. So why does Devanagari choose not to break before an -aa and other scripts do?

There is a huge difference between Devanagari and South East Asian scripts and that is that in Devanagari, a syllable may only contain one vowel character. In SEAsian scripts, vowels are built up from component vowels (each being a vowel in its own right) and therefore a syllable may contain a sequence of vowel characters. [...] But in general SEAsian script users see their vowels as made up of components and not as a single unity. This is why the Burmese want to type their -e vowel before the base consonant whereas they are happy to write their -r medial after (even though it renders in front).

I am therefore not at all surprised that there is an intuitive desire in Ahom to treat spacing vowels as individual letters.

This raises another important point which is premature standardisation. Nobody has any trouble with the idea that a standards body has to choose one choice over another. The problem is if they then say that that choice has to be correctly the first time and may not be changed any time after. [...] This group has probably never thought about what it wants behaviourally with regard to spacing vowels. Looking at Ahom, users may have a hard time deciding what is spacing and what isn't. The Ahom are not in a position to change the Unicode meta model and so they fit themselves as best they can. It is too early in the lifecycle of the Ahom script on computers for users to have strong opinions on this, and ideally, different implementations should allow different behaviours in this area so that users can experience both and build up a preference. As it is, we are sitting in 3 other countries to the Ahom area, deciding what will be the standard behaviour for a group that doesn't even know what it thinks on the topic. This does not bode well in the long term.

"But we have to do something!" I hear you cry. And I sympathise with that need to. Hence the answer you have been given (to allow a cursor before a spacing vowel). The choice also maximises user flexibility. Having to press an arrow key twice when you would prefer once, is far more preferable to not being able to get the cursor where you want it. It is unfortunate that our technical models do not accord the -e vowel the same behavioural characteristic.

My apologies for a rambling excursus over what was asked as a simple, straightforward question. The problem is that in SEAsian scripts, there is no such thing as a simple question.

GB,
Martin

> Thanks for including me on this email; I've read it through (although the link you gave from Neal Johnson didn't work for me) and I've been thinking about it.
>
> Without being completely sure why I think this, I feel it would be better if the 'SpacingMark property' was present, in other words that the Tai Ahom AA behaved like Thai script, not like Nagari script.
>
> There are two reasons why I think this would be best
>

> 1) I think there could be cases of combined syllables that are analogous to the combination te yau that is made with the last listed variant form in the original proposal that Martin and I wrote, which is “actually two characters AHOM LETTER TA (U+11704) AHOM LETTER JA (U+11709)”, except that usually the alternative TA (U+11705) is used
> Thus one can get AHOM VOWEL SIGN E (U+11726) combining with AHOM LETTER ALTERNATIVE TA (U+11705) AHOM LETTER JA (U+11709) and then AHOM VOWEL SIGN AW (U+11727). And it is pronounced te yau (where au rhymes with ‘cow’ and ‘now’).
>
> To me this possibility indicates a certain independence of the vowel from the consonant.
>
> 2) I’m not completely sure if this is the case, but we do occasionally need to write consonant + AHOM VOWEL SIGN AA (U+11721) + AHOM SIGN KILLER (U+1172B). I suspect this might work better if the ‘Spacing Mark Property’ is present.
>
> But I’m not really very expert in all this so not quite sure.
>
> Stephen
>
> From: Peter Constable
> Sent: Saturday, 22 May 2021 5:22 AM
> To: Stephen Morey; Martin Hosken
> Cc: Deborah Anderson
>
> Stephen, Martin:
>
> There was feedback on the Unicode Text Segmentation specification (UAX #29) regarding the Ahom vowel signs A and AA (U+11720, U+11721) because there’s a discrepancy between something in that spec and in a related character properties file, GraphemeBreakProperty.txt. (See the feedback from Neal Johnson in feedback on Public Review Issue 417<https://www.unicode.org/review/pri417/)>.)
>
> The specific issue has to do with whether these vowel signs have the SpacingMark property: the GraphemeBreakProperty.txt file assigns them this property, but the text of UAX #29 says they are excluded from SpacingMark.
>
> The significance of the SpacingMark property for these signs has to do primarily with the expected editing behaviour: Are the signs treated like separately-editable elements, or are they joined to their base? Can they be selected separately when editing? Can the insertion point go between them and the base?
>
> This can be illustrated by example using some other scripts, such as Devanagari versus Thai: here are a couple of hypothetical syllable sequences from these scripts. In each case, try to select the vowel mark /aa/ on the right, or (after starting a reply so you can edit) try setting your cursor to the position between the consonant /k/ (in the middle) and the vowel mark /aa/.
>
> कि
> โก
>
> You can do it in the Thai case, but not the Devanagari case.
>

> This is because they have different properties that affect grapheme segmentation. Combining marks in general will be part of the same grapheme cluster as their base, and the implication is that they cannot be selected separately when editing—e.g., you can't select just the acute accent in "ó". In the Devanagari case, the /aa/ vowel is a combining mark, and it can't be selected separately because of the SpacingMark property which binds it in a grapheme cluster to the base /k/.
>
> In Thai script, the vowel /aa/ is not a combining mark at all—Thai and Lao don't follow the usual Indic encoding models in this regard—and so it is not SpacingMark = True; hence it is selectable.
>
> In SE Asia scripts, there's a general preference to have the same behaviour as in Thai for post-base vowel signs like /aa/. And so in UAX29, the definition for the SpacingMark property has some exceptions: characters like 102C "◌ာ" MYANMAR VOWEL SIGN AA that would otherwise be part of that property by virtue of their General Category property (Mc) but are explicitly listed as exceptions so that they can behave in editing the same as the Thai post-base vowel sign.
>
> The open question is whether Ahom post-base vowel signs A and AA should be considered SpacingMark = True (inherited by virtual of their General Category property), or should be also be exceptions. If they are exceptions, that would imply that they are expected to behave in editing like the Thai case. If they are not exceptions, that would imply that they are expected to behave in editing like Devanagari /aa/.
>
> So, which would be the preferred editing behaviour for these Ahom vowel signs?
>
> Thanks
> Peter Constable

PRI #423: Proposed Update UTS #39 Unicode Security Mechanisms

https://www.unicode.org/review/pri423/

PRI423a: Incorrect Identifier Type for Khmer

Recommended UTC actions

1. Change U+17CB, U+17CC, U+17CD, U+17D0 to Identifier_Type=Recommended, for Unicode 14.
2. Action item for Mark Davis and Markus Scherer: Change U+17CB, U+17CC, U+17CD, U+17D0 to Identifier_Type=Recommended, for Unicode 14.

Feedback (verbatim)

Date/Time: Sun May 23 12:29:13 CDT 2021
Name: asmus
Report Type: Public Review Issue
Opt Subject: PRI 423: Incorrect Identifier Type for Khmer

The Identifier_Type values for certain Khmer characters appear questionable
in light of their support for domain names in the DNS Root Zone.

U+17CB	◌	Khmer	KHMER SIGN BANTOC
U+17CC	◌	Khmer	KHMER SIGN ROBAT
U+17CD	◌	Khmer	KHMER SIGN TOANDAKHIAT
U+17D0	◌	Khmer	KHMER SIGN SAMYOK SANNYA

See: Root Zone Label Generation Rules for the Khmer Script (und-Khmr)
on icann.org and documents cited therein

Proposal for Khmer Script Root Zone LGR, 15 August 2016,
https://www.icann.org/en/system/files/files/proposal-khmer-lgr-15aug16-en.pdf

The document referenced for inclusion of these characters cites the following source, which seems to strongly contradict a "technical use" classification

[204] PRIMARY SCHOOL GRADE 1, MOEYS, ISBN 9-789-995-001-674, Publication 2015, Figure 2

These are also supported in ICANN's Reference LGR for the Second Level.

https://www.icann.org/sites/default/files/packages/lgr/lgr-second-level-khmer-script-15dec20-en.html

Background information / discussion

https://www.unicode.org/Public/security/14.0.0/IdentifierType.txt (pre-release)

```
17CB..17D0    ; Technical                     # 3.0    [6] KHMER SIGN BANTOC..KHMER SIGN
SAMYOK SANNYA
```

Asmus recommends Identifier_Type=Recommended for these characters.

U+17CE, U+17CF are pronunciation markers, and should remain as Technical for now.

PRI #427: Proposed Update UTS #18, Unicode Regular Expressions

https://www.unicode.org/review/pri427/

PRI427a: Examples out of line with UCD

Recommended UTC actions

1. Action item for Mark Davis and the editorial committee: Add a note in UTS #18 like the one in UAX #14 about data files being normative, and examples representing a snapshot, not latest data; for Unicode 14.
2. Action item for Mark Davis and Markus Scherer: Review the UTS #18 examples listed in L2/21-126 item PRI427a for correctness and readability; for Unicode 14.

Feedback (verbatim)

Date/Time: Wed Jun 16 23:40:43 CDT 2021

Name: Wang Yifan
Report Type: Error Report
Opt Subject: PRI #427: Examples out of line with UCD

Some examples currently given in UTS #18 seem to have been either
wrong or outdated.

In the table showing expressions related to hiragana under Section 1.2.6:

Expression | Contents of Set
\p{sc=Hira} | [あ-ゖ ゝ-ゟ □🈀]
\p{scx=Hira} | [、-〃〆〈-】〓-〟〰-〵〷〼-〿 ぁ-ゖ □-゠・ー｜-人ー-〇㈠-㉃㊀-㊰1月-12月0点-24点 ㍻-㍿1日-31日﹅﹆｡-･ｰﾞﾟ□🈀]

But neither line reflects the current state of set in U13.0 or the
proposed U14.0. Moreover, it contains some unneeded spaces.

They should look like (I'm just writing manually; please generate from
data files for accuracy):

Expression | Contents of Set
\p{sc=Hira} | [あ-ゖ ゝ-ゟ □-□□-□🈀]
\p{scx=Hira} | [、-〃〈-】〓-〟〰-〵〷〼〽ぁ-ゖ□-゠・ー﹅﹆｡-･ｰﾞﾟ□-□□-□🈀]

Also, the second (currently first) table under Section 1.1:

Syntax | Matches
[\u{3040}-\u{309F} \u{30FC}] | Hiragana characters, plus prolonged sound sign

The description is not enough accurate as well as misleading as of today.
It should say "Hiragana block code points" instead of "Hiragana characters"
for maximal accuracy.

Though I only spotted issues around Hiragana because it caught sight of me
intuitively, there could be more examples needing maintenance.

Background information / discussion

Markus:

- We generally don't update examples in documents to show complete sets of characters according to each latest version of Unicode.
- There is one space in the middle of "ぁ-ゖ □-゠", probably to avoid the combining sound mark being displayed on top of the Small Ke. We could document the use of spaces, or replace the space + sound mark with "\u3099".
- The \p{scx=Hira} set appears to contain characters with sc=Hani and whose scx do not include Hira. Should check, and compare with \p{scx=Hani}.
- It is correct that \u{3040}-\u{309F} is the entire Hiragana block, including some unassigned code points.

PRI #429: Proposed Update UTS #46, Unicode IDNA Compatibility Processing

https://www.unicode.org/review/pri429/

PRI429a: UTS #46 IDNA issue report

Recommended UTC actions

1. Action item for Mark Davis, Michel Suignard, and Markus Scherer: Check the IdnaTestV2.txt bug report in L2/21-126 item PRI429a, and fix it if appropriate; for Unicode 14.
2. Action item for Mark Davis, Michel Suignard, and Markus Scherer: Investigate the proposals in L2/21-126 item PRI429a; take into consideration the relationship of UTS #46 with IDNA2008; for Unicode 15.

Feedback (verbatim)

Date/Time: Sat May 22 04:42:52 CDT 2021
Name: Timothy Gu
Report Type: Error Report
Opt Subject: UTS #46 IDNA issue report

Hello,

I'm looking to fully implement UTS #46 in Google Chrome in alignment with other web browsers and the WHATWG URL Standard. Unfortunately, while doing so, we have discovered the following issues with UTS #46. We also attached a proposed solution for each issue listed.

Forbid double-encoded xn-- even with CheckHyphens=false

The CheckHyphens boolean flag was introduced in version 10.0.0 of UTS #46. It loosens the DNS restriction of having -- in the third and fourth places of a domain label, in order to support certain existing deployed content. However, the introduced flag has a defect: it allows double-encoded IDNA labels to be considered valid.

Here's one example of how this is bad. Consider the domain label "xn--xn---epa". Assuming CheckHyphens=false, upon applying ToUnicode, it would get converted to "xn--é" without any errors. However, this conversion would not round trip, since applying ToASCII to "xn--é" would produce a failure value.

We propose the following fix. In Section 4.1, Validity Criteria, insert the following item after criterion 3:

> If not CheckHyphens, the label must not begin with “xn--”.

Provide a mode to keep ASCII labels identical

Under the current UTS #46 processing, the label "xn--a" is considered invalid since it suffers from a Punycode decoding error. Yet, existing implementations universally accept "xn--a.com" as a valid domain for lookup. In order to reflect reality, UTS #46 needs to make provisions for such "ASCII fast path." However, to prevent roundtripping bugs, there is also a need to maintain the same validity status for equivalent A- and U-labels. (E.g., "xn--a-ecp.com" should continue to be invalid, just like how "a1..com" is invalid.)

My proposal is to introduce a new boolean flag IgnoreInvalidPunycode. The algorithm in Section 4, Processing should then be amended as follows. Replace step 4.1.1 which currently says:

> Attempt to convert the rest of the label to Unicode according to
> Punycode [RFC3492]. If that conversion fails, record that there was an
> error, and continue with the next label. Otherwise replace the
> original label in the string by the results of the conversion.

with the two steps

> **If the label contains any non-ASCII code point (i.e., a code point
> greater than U+007F), record there was an error, and continue with the
> next label.**
>
> Attempt to convert the rest of the label to Unicode according to
> Punycode [RFC3492]. If that conversion fails, **and if not
> IgnoreInvalidPunycode,** record that there was an error, and continue
> with the next label. Otherwise replace the original label in the
> string by the results of the conversion.

(Additions are surrounded by two asterisks.)

These changes would continue to make ToASCII return failure for labels such as "xn--é" and "xn--a-ecp", but keep "xn--a" as is. Additionally, since the rule operates on the basis of labels, it would be okay with "xn--a.xn--nxa" ("xn--a.β" in Unicode), matching existing real-world implementations.

As a reference, here is a partial list of important implementations that currently support IDNA but also allow "xn--a.com":

- curl
- Firefox Browser
- GNU Wget
- Go net/http

- Google Chrome
- Safari

IdnaTestV2.txt issue

The IdnaTestV2.txt that came with UTS #46 Version 13.0.0 has a minor error. ToUnicode for "xn--mbm8237g..xn--7-7hf" is marked as failure with reason "V6", but this should not be the case. (This domain appears _twice_ around line 3070.)

The cause appears to be the lack of update for version 13.0.0. The problematic domain contains the code point U+18C4E. While this code point is "disallowed" in Unicode 12.0.0's IdnaMappingTable.txt (line 6397), it is "valid" under Unicode 13.0.0's (line 6450).

(Note, however, a similar domain "xn--mbm8237g.xn--7-7hf1526p" is correct in the file and should remain forbidden. This is since it additionally has the code point U+FE12, which is "disallowed" in both 12.0.0 and 13.0.0.)

ICU support

Finally, I request that the ICU libraries add support for the two features mentioned above. If ICU is outside the purview of this committee, please kindly let me know so that I can forward the request to the right people.

For the first issue, ICU4C does not make it possible to distinguish labels of type "xn--xn---epa" versus "ab--cde": both return UIDNA_ERROR_HYPHEN_3_4. Some way of forbidding the latter but allowing the former would be useful.

For the second issue, additional changes may be needed as ICU returns "xn--a" with a U+FFFD at the end for "xn--a". We would want to keep the original label unchanged for ASCII labels with Punycode decoding errors. A separate uidna_openUTS46() option may be necessary.

Similar changes would probably be needed for ICU4J. ICU4X does not plan to support IDNA.

Background information / discussion

Mark: I just scanned this quickly, but on first glance looks like good changes

PRI429b: Meta-issue: Active ownership & review

Recommended UTC actions

1. FYI: We have concerns about inactive ownership, and lack of review of data used in significant implementations and products.

Background information / discussion

We discussed the issue of active ownership and review of some of the Unicode “projects” and especially their associated data files. For example, for security (UTS #39) and IDNA (UTS #46) which are used in significant implementations, we use existing code to generate data files, including for which characters and scripts can (IDNA) and should (security) be used in domain names. However, there is no active owner on the Unicode side that reviews the generated data in detail, there is generally little feedback during the beta period on the Identifier_Type for new and existing characters, and there is generally no feedback on changes in and additions to the IdnaMappingTable.

PRI429c: additional data files for IDNA2008 for easy review by IETF

Recommended UTC actions

1. Action item for Ken Whistler and Asmus Freytag: Create additional derived data files showing IDNA2008 category values for easy comparisons with those published by the IETF, for Unicode 15. See L2/21-126 item PRI429c.

Feedback (verbatim)

Proposal from Asmus Freytag, 2021-jul-16:

1. Create a derived property file that captures precisely the IDNA2008 calculated category values.
2. Create an “other_IDNA2008” property to hold override values like CONTEXTO, etc.

Details:

Several *people, including people generally knowledgeable in IDNs or Unicode, have reported that they have difficulty in extracting the IDNA 2008 related status values from the existing data tables for UTS#46. This situation may be common.*

IETF runs a review process under RFC5892 as updated by RFC8753 to check whether any Unicode property values have changed, or whether any calculated properties for new characters are inappropriate. In both cases, the process allows for exceptional values to be applied.

Because IDNA2008 values are derived algorithmically from Unicode properties, this is just a restatement of existing information. However, any exceptional overrides applied in RFC5892 or later as part of IETF review should be cataloged in an Other_IDNA2008 property that is used by the algorithm to override these values.

RFC5892 considers the calculation of values normative; however, exceptions can in principle be declared at any time and not in sync with Unicode versions. To handle this, the status of the derived property should be informative, with a stipulation that in case of difference, the values arrived based on the IDNA2008 RFCs are considered the correct ones. As for any published exceptions: Unicode would pick these up in the next release of the derived data file. In essence, the derived data file would represent a snapshot at the time of calculation. That should be in keeping with the way IETF approaches their own tables registered at IANA (but which are usually a version or more behind the latest Unicode version). The review for Unicode 11.0. And 12.0 is still in the internet draft stage and not an RFC yet.